Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная работа № 2**

Программирование RISC-V

по дисциплине «Низкоуровневое программирование»

Выполнил

студент гр. 3530901/90004

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гасилов Д.О.

(подпись)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург   
2021

**Задача**

1. Разработать программу на языке ассемблера RISC-V, реализующую определенную вариантом задания функциональность, отладить программу в симуляторе VSim/Jupiter. Массив (массивы) данных и другие параметры (преобразуемое число, длина массива, параметр статистики и пр.) располагаются в памяти по фиксированным адресам

2. Выделить определенную вариантом задания функциональность в подпрограмму, организованную в соответствии с ABI, разработать использующую ее тестовую программу. Адрес обрабатываемого массива данных и другие значения передавать через параметры подпрограммы в соответствии с ABI. Тестовая программа должна состоять из инициализирующего кода, кода завершения, подпрограммы main и тестируемой подпрограммы.

**Вариант задания**

Вариант: 9 - Расчет биномиальных коэффициентов для данного показателя по треугольнику Паскаля.

**Выполнение работы**  
Изначально мы имеем массив под меткой «array», состоящий из двух единиц (коэффициенты при показателе равном единице)

Показатель – под меткой «pokaz». В качестве пример рассмотрим показатель 5

Суть алгоритма заключается в том, чтобы проходить по массиву, складывать значения двух соседних элементов и перезаписывать их и увеличивать массив в размерах. По массиву мы проходим столько раз, сколько необходимо для вычисления коэффициентов для заданного показателя.

Код программы для задачи 1 с комментариями:  
  
.text

start:

.globl start

la a2, pokaz #}

lw a2, 0(a2) #} a2 = <заданный показатель>

li a3, 1 # a3 = 1

li a4, 0 # a4 = 0

la a5, array # a5 = <адрес 0-го элемента массива>

loop1:

bgeu a3, a2, loop\_exit1 # if( a3 >= a2 ) goto loop\_exit1

addi a3, a3, 1 # a3 += 1

slli a6, a4, 2 # a6 = a2 << 2 = a3 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a3 \* 4

addi a7, a6, 4 # a7 = a6 + 4

lw t1, 0(a7) # t1 = array[i+1]

lw t0, 0(a6) # t0 = array[i]

add t0, t1, t0 # t0 = array[0] + array[1]

loop2:

bgeu a4, a2, loop\_exit2 # if( a4 >= a2 ) goto loop\_exit2

addi a4, a4, 1 # a4 += 1

slli a6, a4, 2 # a6 = a4 << 2 = a4 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a3 \* 4

addi a7, a6, 4 # a7 = a6 + 4

lw t1, 0(a6) # t1 = array[i]

sw t0, 0(a6) # array[i] = t0

lw t0, 0(a7) # t0 = array[i+1]

add t0, t1, t0 # t0 = array[i+1] + t1

jal zero, loop2 # goto loop1

loop\_exit2:

li a4, 0 # a4 = 0

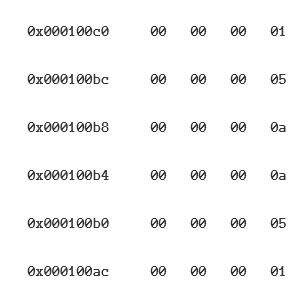
jal zero, loop1 # goto loop1

loop\_exit1:

li a0, 10 # x10 = 10

li a1, 0 # x11 = 0

ecall # ecall при значении x10 = 10 => останов симулятора

Результат работы программы  
 

Аналогичный алгоритм используется для задачи 2. Так же разработана вызывающая ее тестовую программа. Адрес обрабатываемого массива данных и показатель передаются через регистр a0 и a1 соответственно.  
  
Код программы задачи 2 состоит из трёх файлов ниже представлены эти файлы с комментариями:

Файл 1

# setup.s

.text

start:

.globl start

call main

finish:

mv a1, a0 # a1 = a0

li a0, 17 # a0 = 17

ecall # выход с кодом завершения

Файл2  
# coeffs.s

.text

coeffs:

.globl coeffs

li a3, 1 # a3 = 0

li a4, 0 # a4 = 0

loop1:

bgeu a3, a2, loop\_exit1 # if( a3 >= a2 ) goto loop\_exit1

addi a3, a3, 1 # a3 += 1

slli a6, a4, 2 # a6 = a2 << 2 = a3 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a3 \* 4

addi a7, a6, 4 # a7 = a6 + 4

lw t1, 0(a7) # t1 = array[i+1]

lw t0, 0(a6) # t0 = array[i]

add t0, t1, t0 # t0 = array[0] + array[1]

loop2:

bgeu a4, a2, loop\_exit2 # if( a4 >= a2 ) goto loop\_exit1

addi a4, a4, 1 # a4 += 1

slli a6, a4, 2 # a6 = a4 << 2 = a4 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a3 \* 4

addi a7, a6, 4 # a6 = a5 + 4

lw t1, 0(a6) # t1 = array[i]

sw t0, 0(a6) # array[i] = t0

lw t0, 0(a7) # t0 = array[i+1]

add t0, t1, t0 # t0 = array[i+1] + t1

jal zero, loop2 # goto loop1

loop\_exit2:

li a4, 0 # a4 = 0

jal zero, loop1 # goto loop1

loop\_exit1:

ret li a7, 0 # a7 = 0

slli a5, a7, 2 # a5 = a2 << 2 = a2 \* 4

add a5, a0, a5 # a5 = a0 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a6) # t1 = array[i+1]

lw t0, 0(a5) # t0 = array[i]

add t0, t1, t0 # t0 = array[0] + array[1]

loop2:

bgeu a7, a2, loop\_exit2 # if( a2 >= a3 ) goto loop\_exit

addi a7, a7, 1 # a7 += 1

slli a5, a7, 2 # a5 = a7 << 2 = a7 \* 4

add a5, a0, a5 # a5 = a0 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a5) # t1 = array[i]

sw t0, 0(a5) # array[i] = t0

lw t0, 0(a6) # t0 = array[i+1]

add t0, t1, t0 # t0 = array[i+1] + t1

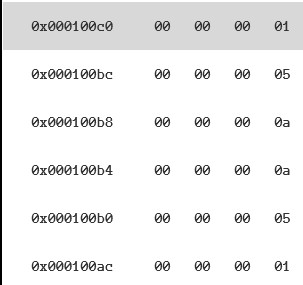
jal zero, loop2 # goto loop

loop\_exit2:

jal zero, loop1 # goto loop

loop\_exit1:

ret

Результат работы вызванной подпрограммы:  


**Вывод**

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, выполняющая расчет биномиальных коэффициентов для данного показателя по треугольнику Паскаля. Так же была создана тестовая программа, вызывающая подпрограмму, работающую с загруженными в нее данными о показателе числа и адресе массива